Hệ điều hành thời gian thực

Tiếp cận đa tác vụ

- Tiếp cận chương trình đơn: một tác vụ (luồng)
- Tiếp cận gần/tiếp cận sau: 2 tác vụ
- Tổng quát hóa, đa tác vụ là:
  - Cũng gọi là các quy trình, luồng
  - Mỗi tác vụ được xử lý song song
    - Không chiếm toàn bộ tài nguyên xử lý
    - Các tác vụ tương tác đồng thời tới các phần tử bên ngoài
      - Điều khiển bộ cảm biến, điều khiển cơ cấu truyền động thông qua DMA, bộ ngắt, I/O.
    - Thường được xử lý song song
  - Yêu cầu
    - Lập lịch cho các tác vụ
    - Chia xẻ dữ liệu giữa các tác vụ đồng thời

Các dấu hiệu của tác vụ

- Các tác vụ mới có thể khởi động trong khoảng thời gian
  - Theo chu kỳ hoặc không theo chu kỳ
- Tác vụ có thể gồm:
  - Tài nguyên cần thiết
  - Các mức quan trọng
  - Quan hệ ưu tiên
  - Liên kết
  - Giới hạn thời gian
- Trạng thái của tác vụ
  - Hoạt động
  - Chuẩn bị tác vụ - chờ xử lý từ CPU
  - Hạn chế tác vụ - chờ khi có sự kiên khác tác động tới quá trình xử lý
  - Không hoạt động

Quyền ưu tiên

- Không có ưu tiên: tác vụ khi khởi động sẽ chạy đên khi dừng lai hoặc tác động tới một số cổng vào/ra
- Ưu tiên: tác vụ có thể dừng lại để chạy tác vụ khác
  - Chịu sự ưu tiên và sự phức tạp trong thực thi
  - Có quá trình lập lịch và phân tích
  - Trong không ưu tiên, các tác vụ hoạt động do bị ép buộc

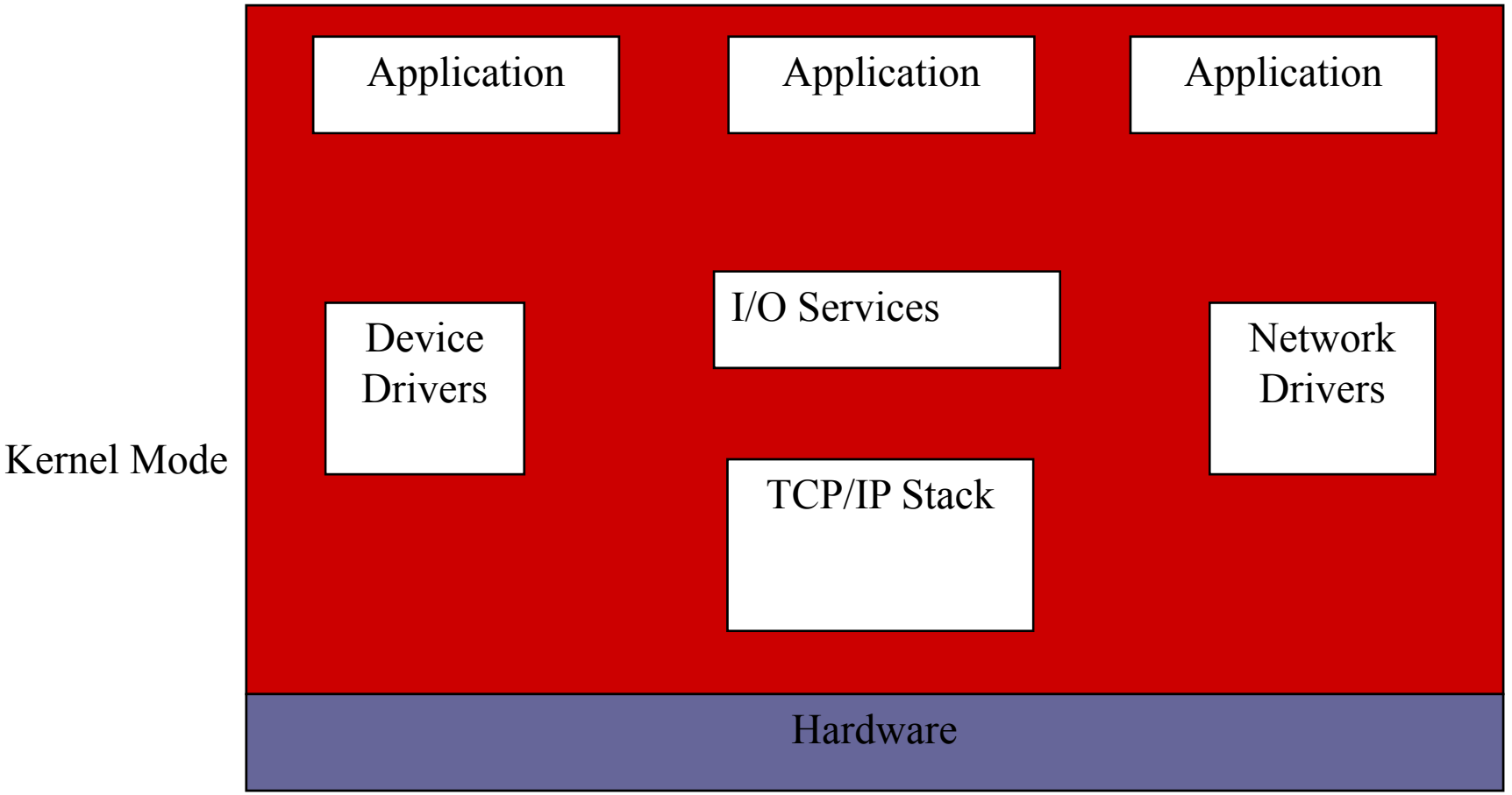
Lập lịch cho đa tác vụ

- Kế hoạch lập lịch động
  - Các tác vụ xảy ra động
  - Quá trình kiểm tra lịch xảy ra mỗi khi có tác vụ động được hoạt động
  - Khi có tác vụ mới, trước khi khởi động
    - Lập lịch hoạt động cho tác vụ trước và sau
    - Nếu quá trình lập lịch có lỗi, có phương án thay đổi
  - Lịch nhận được sẽ quyết định khi khởi tạo
- Kết quả đạt được của lập lịch động
  - Hệ thống cố gắng đạt được thời hạn về thời gian hoạt động
  - Không biết khi nào giới hạn về thời gian xảy ra cho đên khi gặp deadline hoặc tác vụ kết thúc hoạt động

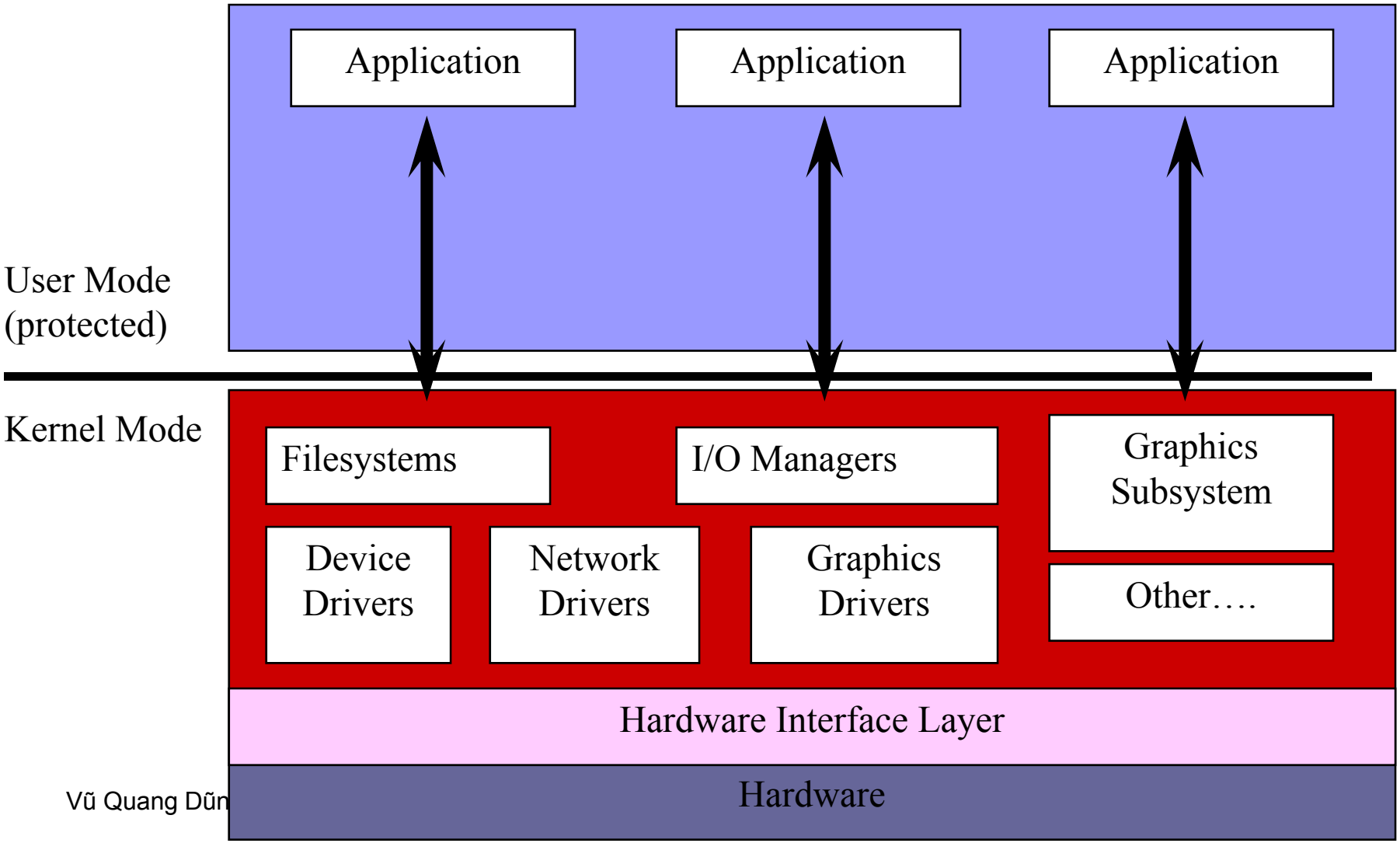
Thực thi trong hệ điều hành thời gian thực

- Nhỏ, nhanh
- Chế độ thời gian thực
- Nghiên cứu hệ điều hành
- Thành phần của ngôn ngữ tạo môi trường
  - Java (embedded real-time Java)
- Nhân đơn hoặc nhân nhỏ tích hợp
- Vấn đề chính
  - Mô hình song song: sự kiện, luồng, lập lịch
  - Mô hình bộ nhớ: Tĩnh và động
  - Các mô hình thành phần khác nếu có

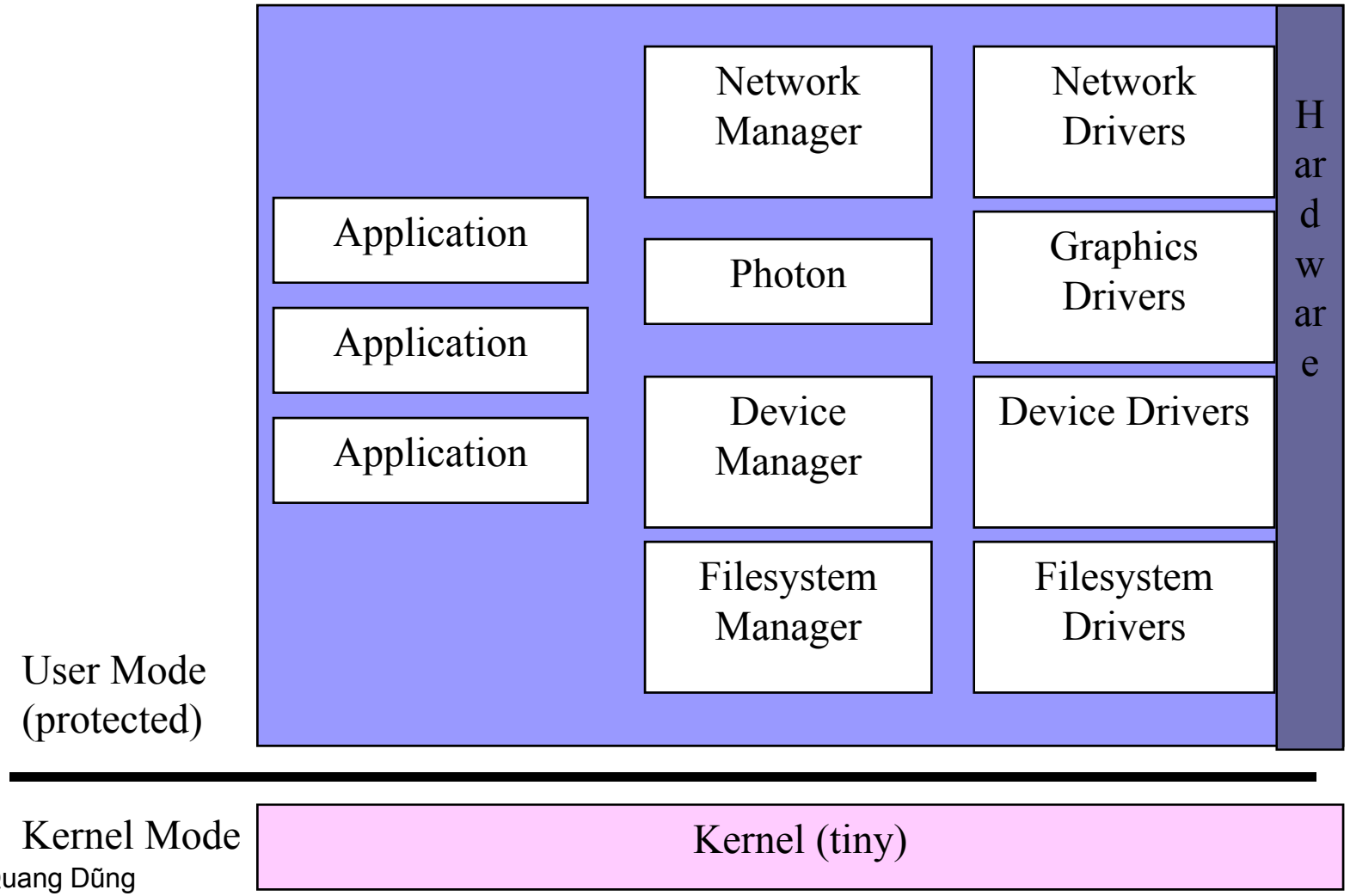
Sự so sánh của các RTOS khác nhau: sự thực thi tuần hoàn



Sự so sánh của các RTOS khác nhau: nhân đơn



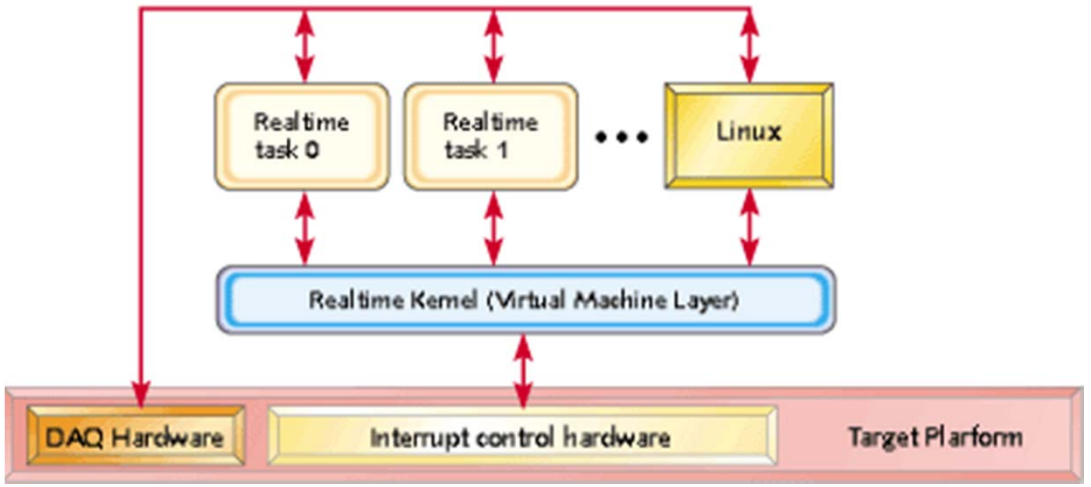
Sự so sánh của các RTOS khác nhau: nhân nhỏ tích hợp



Hệ điều hành thời gian thực linux

- Vi điều khiển:
  - uClinux - Linux nhỏ (nhân < 512KB) với TCP/IP
- Embedded PC
  - RTLinux
  - RTAI (Real-Time Application Interface)
  - TinyOS

RTLinux



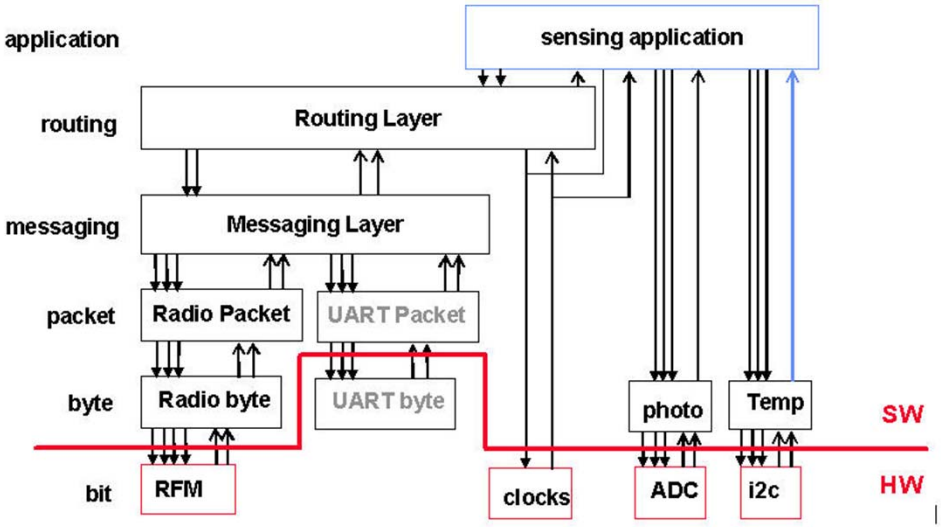
- Nhân Linux hoạt động dưới chế độ ưu tiên nhỏ nhất
  - Nhân Linux hoạt đông theo chế độ ưu tiên
  - Linux nắm bắt và làm việc theo các chế độ ngắt, thông qua các lệnh về ngắt
- Chế độ thời gian thực và xử lý thông tin trong Linux thông qua sự phân chia bộ nhớ và các giao diện của chương trình
- Issues
  - Cơ chế thời gian thực chạy trong cùng một không gian địa chỉ giống như nhân Linux
  - Các device drivers dành cho cơ chế thời gian thực không thể sử dụng chuẩn cơ chế của Linux
  - Không cung cấp các quá trình quản lý tài nguyên
  - Linux drivers được ngắt trực tiếp khi xảy ra sự cố

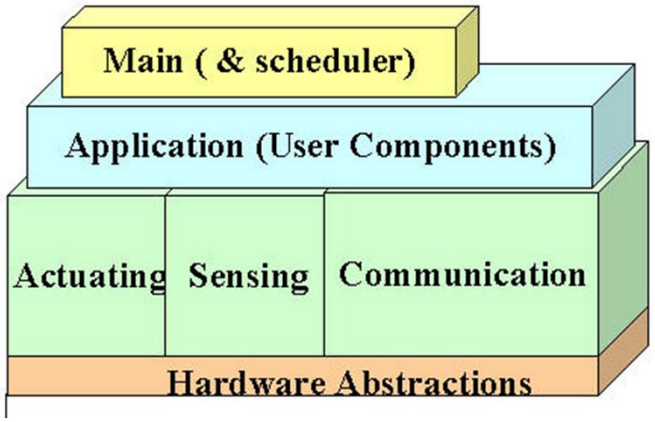
RTAI (Real Time Application Interface)

- Dự án mã nguồn mở xây dựng trên ý tưởng của RTLinux, và được tăng khả năng lên đáng kể
- Linux kernel làm thay đổi trong quá trình nắm bắt hàm ngắt và lập lịch, và được cấu thành bởi
  - Các nền thời gian thực với độ trễ nhỏ và có sự dự đoán cao
  - Và các nền không theo thời gian thực như giao diện người dùng
- Tích hợp giữa lập lịch cho các tác vụ của nhân, luồng xử lý và các tiến trình theo của các tác nhân khác
  - Luồng xử lý được bảo vệ bởi các quá trình trễ trong các đối tượng lập lịch của linux
- Về cơ bản, thì hàm ngắt là quá trình bẫy của các sự kiện bên ngoài và cần thiết cho các tiến trình xử lý khác trong Linux
  - Sử dụng HAL (Hardware Abstraction Layer)
- Về quá trình xử lý động, thì một lớp middleware được dùng để điều khiển các tiến trình gọi thủ tục của các RTAI API

TinyOS

- Không hẳn là một hệ điều hành theo quan niệm truyền thống
- Được cấu trúc giống như một tiến trình xử lý trong hệ điều hành
- Application = Scheduler + Graph of Components
- Mô hình song hành – sự kiện và kiến trúc điều khiển
- Chia xẻ đơn của ngăn xếp
- Lập trình trên NesC, một dạng của C

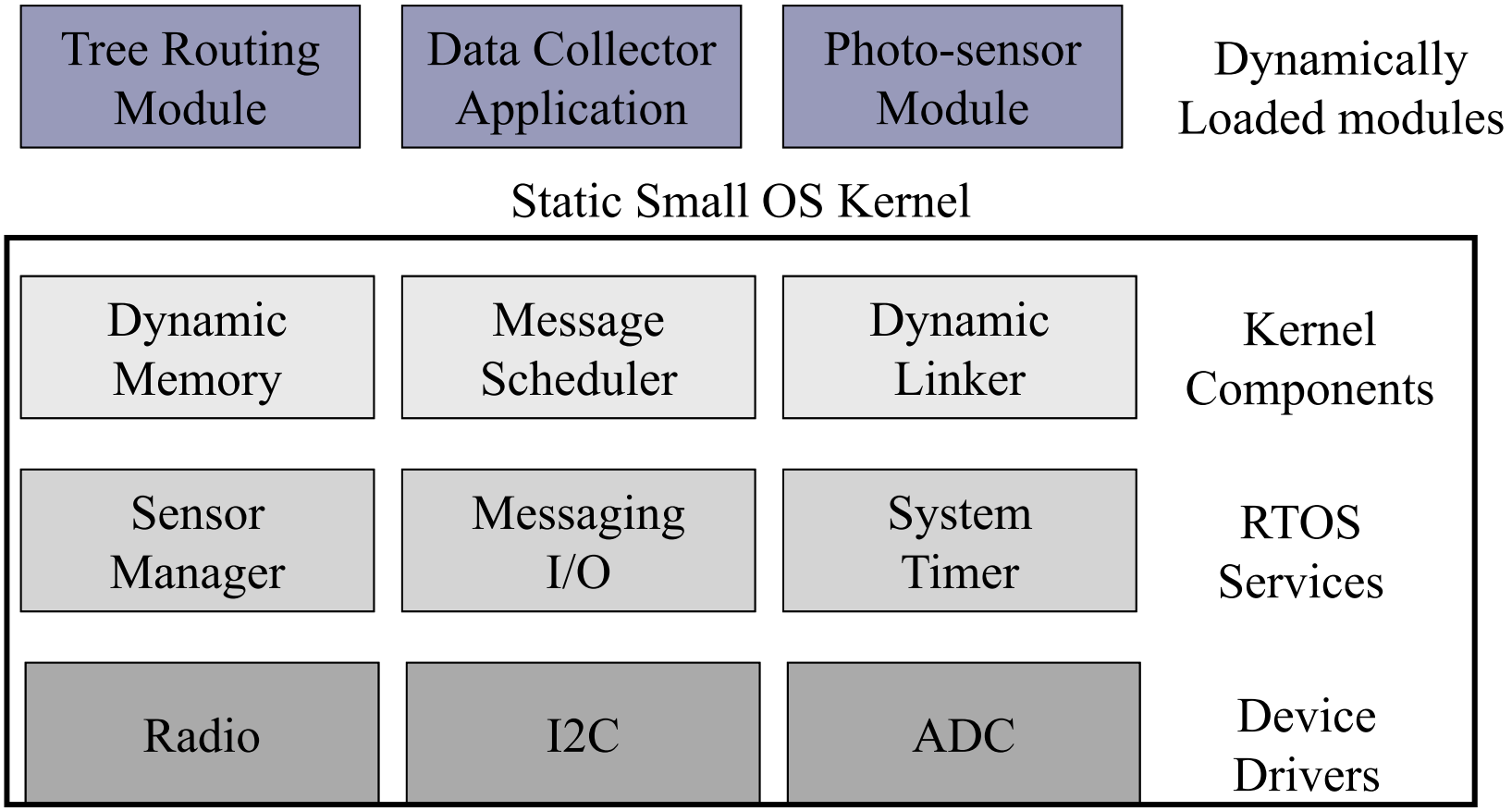




Sự tái cấu hình của phần mềm

- Quá trình cảm nhận và hoạt động của networks đòi hỏi một quá trình liên tục vô hạn
- Sự phát triển của phần mềm là quá trình kế thừa liên tục
  - Tùy biến môi trường hệ thống
  - Quá trình nâng cấp
  - Sửa và loại bỏ lỗi
  - Tái lập các bài toán và tiến trình hệ thống
- Tái lập trình triển khai hệ thống
- Điều khiển sự tái lập trình là bản chất tất yếu của sự bền vững

Kiến trúc hệ thống



Static Kernel

- Trừu tượng hóa phần cứng và các tiến trình
- Thay đổi sau khi triển khai khó và đắt
- Cấu trúc dữ liệu dành cho các module

Dynamic Modules

- Trình điều khiển, protocols, và các chương trình
- Thay đổi sau khi triển khai dễ dàng và rẻ
- Cấu trúc và vị trí độc lập

Embedded Linux

- Bộ công cụ (Tool chains)
- Bootloader
- Xây dựng nhân (Building Linux kernel)
- Điều khiển thiết bị (Linux device driver)
- Giao diện người sử dụng (GUI)

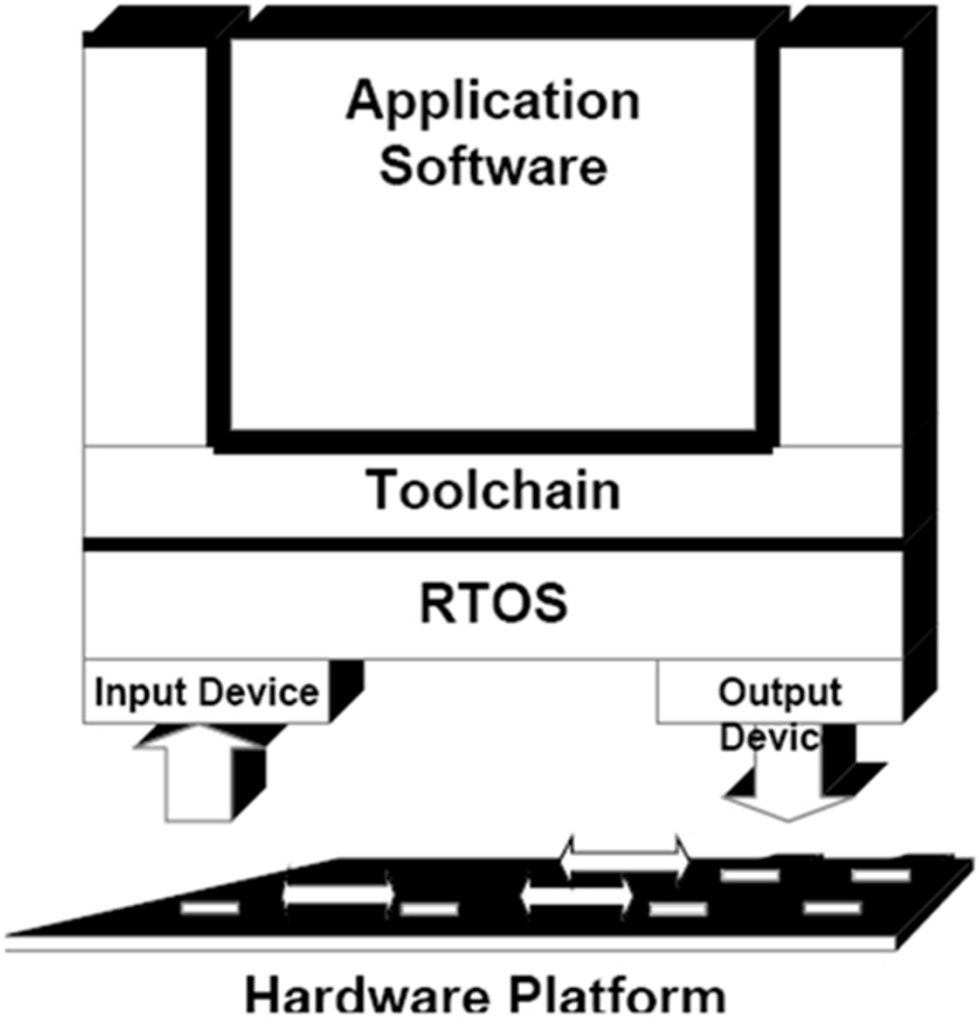
Bộ công cụ

- **Bộ công cụ sử dụng trong phát triển các mục tiêu của phần cứng**
- **Được sử dụng trong phạm vi xây dựng phần mềm trên một hệ thống sẽ được cài đặt hoặc trên một số hệ thống khác**
- **Đước sử dụng với các thiết bị khác để biên dịch, tạo thư viện, ngôn ngữ máy … (compiler, libraries, assembler)**

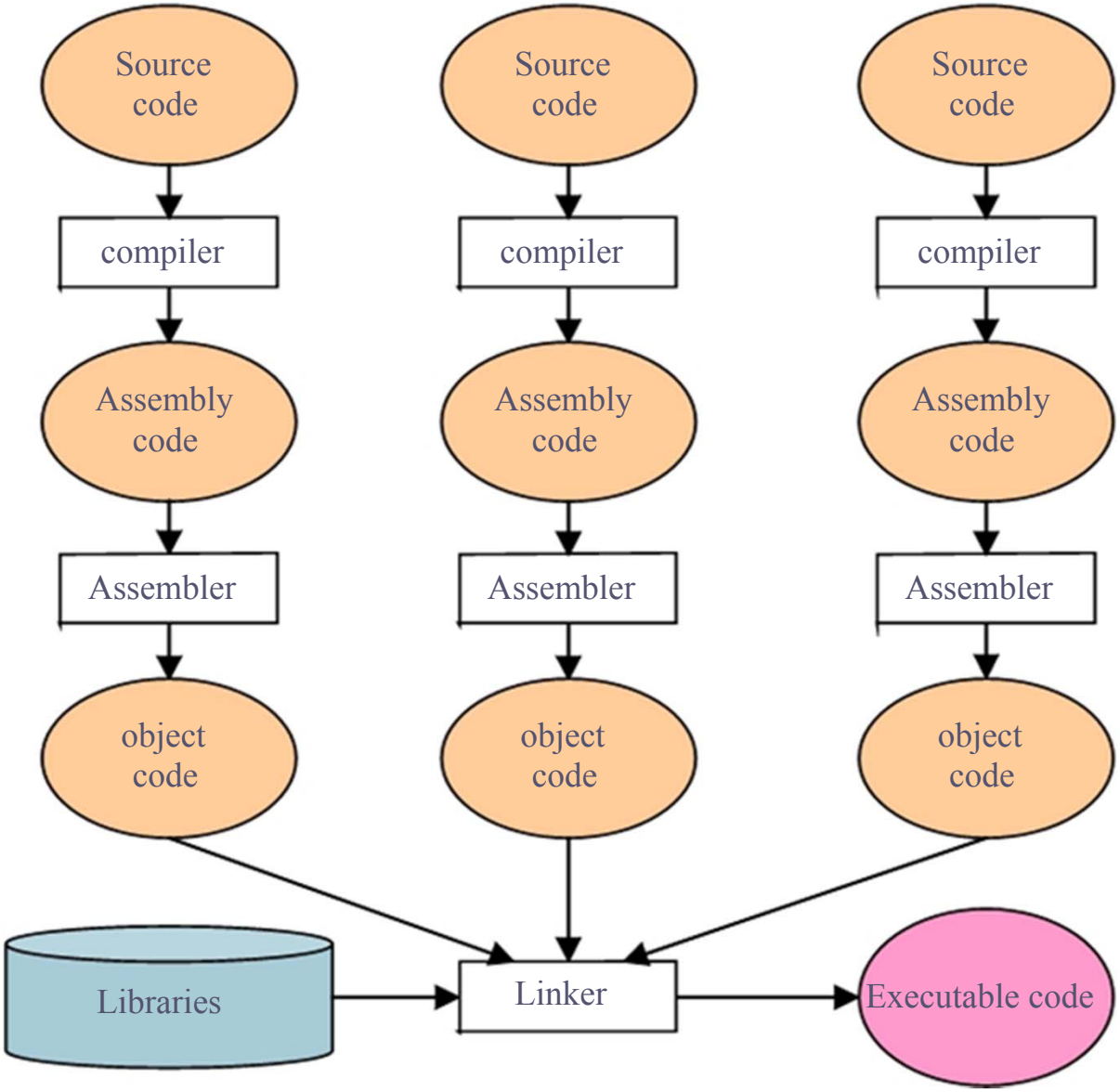
Lợi ích của bộ công cụ

- Bất kỳ dự án nào mà cần thiết trong xử lý nhúng đều cần thiết phải có công cụ phát triển phần mềm
- Đem lại lợi ích trong phát triển hệ thống nhúng mới với bộ công cụ riêng biệt

Vai trò của bộ công cụ



Biểu đồ phát triển



Bộ công cụ GNU

- **GNU là hệ điều hành, giống như Unix (GNU's Not Unix), đã được phát triển trên 20 năm bởi tổ chức phần mềm tự do (FSF – Free Software Foundation)**
- **Phần mềm GNU được biết tới là tính ổn định và tiện lợi**
- **Bộ công cụ GNU là mã nguồn mở**

Thành phần của bộ công cụ GNU

- Biên dịch - gcc
- Assembler - binutils : as
- Liên kết - binutils : ld
- Thư viện - glibc
- Gỡ lỗi - gdb

Gcc – GNU Compiler Collection

- **Gcc được viết hoàn toàn trên ANSI C với sự hỗ trợ biên dịch cho C, C++, Objective C, Java, và Fortran**
- **GCC cung cấp nhiều mức độ kiểm tra lỗi mã nguồn, thông tin về gỡ lỗi và có thể thi hành trên nhiều dạng khác nhau nhằm tối ưu kết quả của mã nguồn**
- **Gcj (GNU Compiler for Java) được tích hợp và hỗ trợ cho mã nguồn Java hoặc Java bytecode**

GNU binutils (GNU Binary Utility)

- **Binutils là tập hợp các công cụ xử lý nhị phân như assembler, liên kết, disassembler … .**
- **Binutils sử dụng để tạo và thao tác với mã nhị phân của file**
- **Thành phần chính gồm**
  - **ld - GNU linker.**
  - **as - GNU assembler.**

Thư viện C trong GNU

- **Bất kỳ hệ thống giống Unix nào đều cần thư viện C, xây dựng các tập lệnh, quản lý tiến trình, bộ nhớ …**
- **Thư viện C được sử dụng như các thư viện C khác trong hệ thống GNU và trong nhân Linux**
- **Thư viện C là sự kết nối trong mã nguồn, và che giấu các chức năng đặc trưng của nhân.**

GDB – GNU Debugger

- **Cho phép xem được tiến trình bên trong của chương trình khác trong thực thi**
- **Cho phép xem được chương trình khác làm gì tại thời điểm dừng khẩn cấp chương trình**
- **GDB có thể chạy trên mọi hệ thống như Unix và có thể trên Windows**

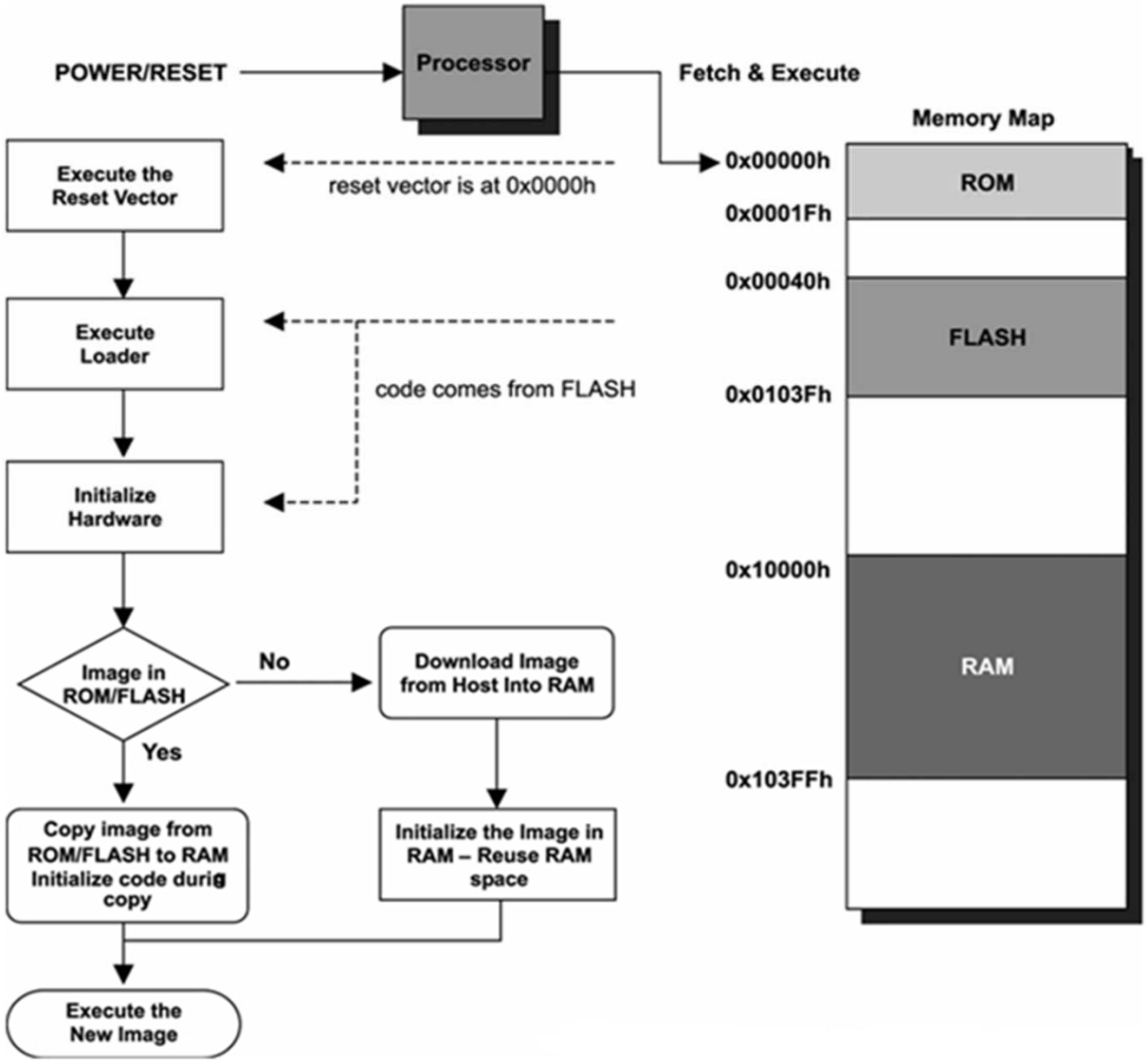
Thành phần ngôn ngữ hỗ trợ của GDB

- C
- C++
- Java
- Objective C

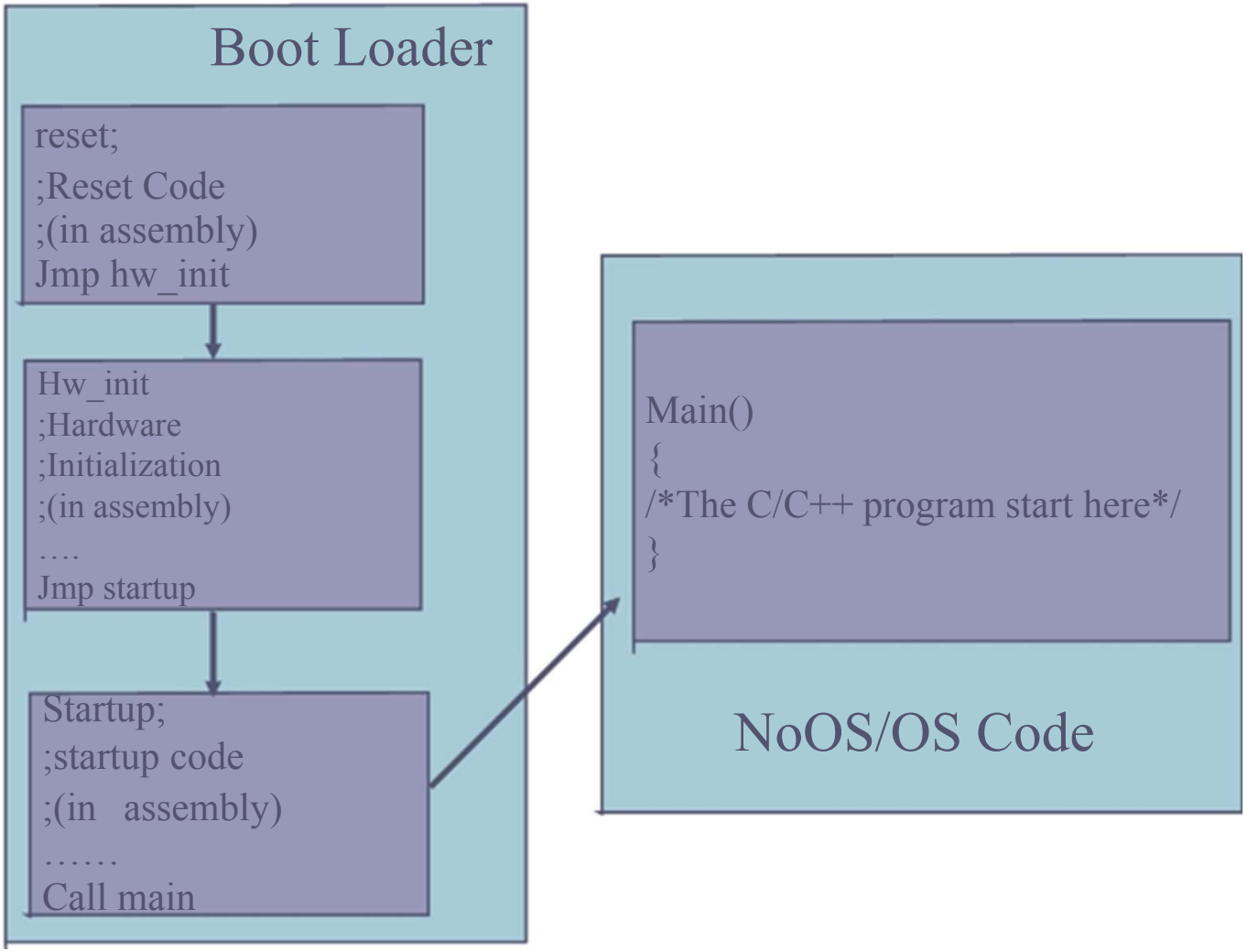
Bootloader

- **Bootloader là thành phần chính, quan trọng khi khởi động nhân**
- **Có nhiều loại bootloaders khác nhau có thể sử dụng với Linux trong nhiều cấu trúc phần cứng**
  - **LOIO**
  - **GRUB**
  - **LinuxBIOS**
  - **Redboot**
  - **U-boot**
- **Bootloader của hệ thống nhúng sẽ làm:**
  - **Khởi tạo phần cứng: bộ vi xử lý và bộ nhớ**
  - **Khởi động nhân và thực thi nhân (trong một số trường hợp cần phải truyền tham số cho nhân linux)**

Ví dụ về bootloader



Tổng quan về Bootloader



Tiến trình Bootloader

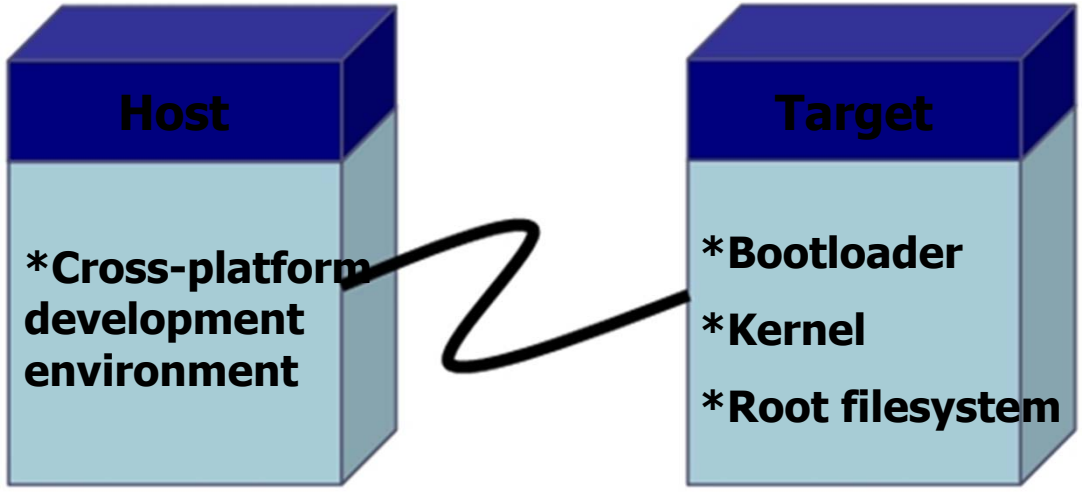
- **Part 1**
  - **Nhảy tới địa chỉ 0x00900000**
- Part 2
  - **Khởi tạo thành phần phần cứng**
  - **Cài đặt chế độ cho CPU**
- **Part 3**
  - **Hủy bỏ tất cả các hàm ngắt**
  - **Copy và khởi tạo dữ liệu từ ROM sang RAM**
  - **Các thành phần không có dữ liệu được khởi tạo với giá trị Zero**
  - **Tính toán không gian cho khởi tạo ngăn xếp**
  - **Khởi tạo con trỏ trong ngăn xếp**
  - **Tạo và khởi tạo bộ nhớ động**
  - **Thực thi các cấu trúc cho các giá trị toàn bộ (C++)**
  - **Cho phép các hàm ngắt**
  - **Gọi hàm main**
- **Part 4**
  - **Chương trình mã C/C++**

Xây dựng nhân Linux

- **Có 3 kiến trúc khác nhau về host/target được sử dụng cho phát triển hệ thống linux nhúng:**
  - **Cài đặt liên kết**
  - **Cài đặt thiết bị lưu trữ di động**
  - **Cài đặt tự động**
- **Quá trình cài đặt có thể liên quan tới một hoặc nhiều thành phần, phụ thuộc vào yêu cầu và phương pháp phát triển hệ thống**

Cài đặt liên kết

- **Đích và nguồn được liên kết cố định với nhau sử dụng liên kết vật lý**
  - Liên kết đó co thể thông qua cổng serial hoặc ethernet
  - Tất cả dữ liệu được chuyển qua liên kết

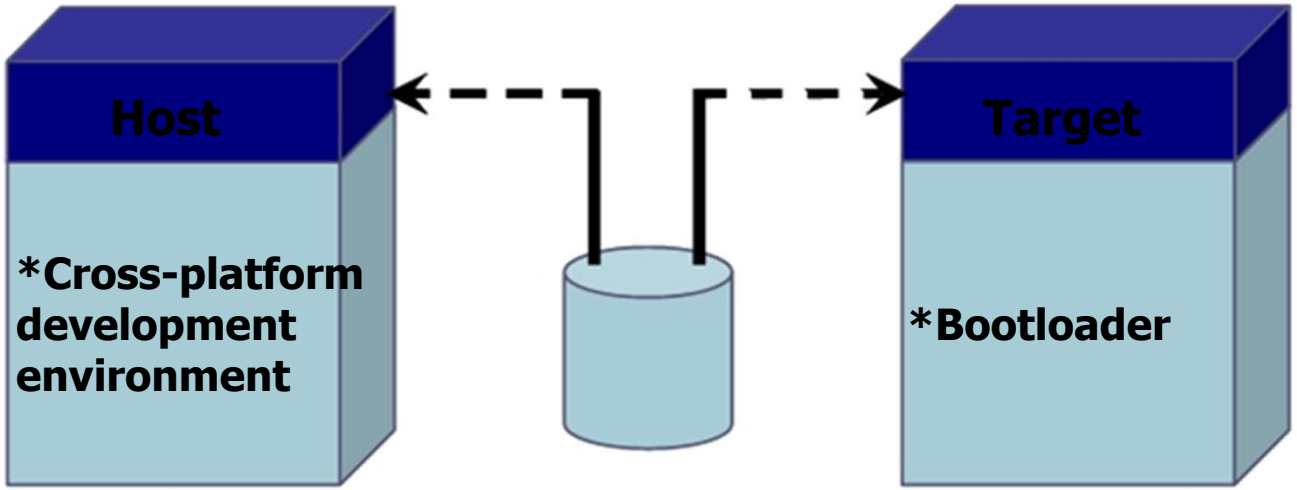


Cài đặt liên kết (tiếp)

- **Nhân được trung chuyển qua ftp cân bằng (TFTP)**
- **Hệ thống file gốc có thể là NFS thay vì là nơi lưu trữ tại nguồn**
- **Liên kết vật lý có thể sử dụng cho mục đích gỡ lỗi**
  - **Nhiều hệ thống nhúng cung cấp cả 2 loại liên kết qua Ethernet và RS232**

Cài đặt thiết bị lưu trữ di động

- **Thiết bị lưu trữ được viết bởi nguồn và sẽ chuyển sang đích, và sử dụng làm thiết bị khởi động**

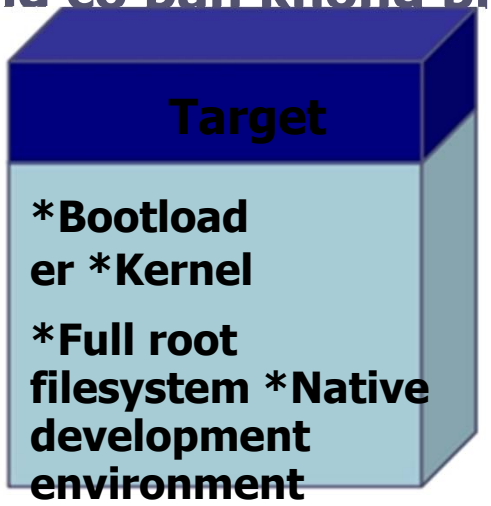


Cài đặt thiết bị lưu trữ di động (tiếp)

- **Đích chứa hệ thống khởi động nhỏ nhất**
  - **Các phần còn lại được lưu trữ tại thiết bị lưu trữ như CompactFlash IDE và một số loại khác**
- **Thay vi dùng chíp flash cố định, nguồn có thể chứa socket mà các chip flash có thể dễ dàng gắn vào và tháo ra**
  - **Chíp sẽ được lập trình bằng flash programmer tại nguồn và sẽ gắn vào socket tại đích**
- **Cài đặt này được sử dụng nhiều trong các giai đoạn đầu của phát triển hệ thống nhúng**

Cài đặt tự động

- **Đích có thể tự chứa hệ thống phát triển và bao gồm tất cả các phần mềm khởi động, điều khiển và phát triển các phần mềm phụ**
  - **Cài đặt này thường sử dụng tại các máy workstation, loại trừ các phần cứng cơ bản không phải của máy trạm mà la thuộc hệ thốn**



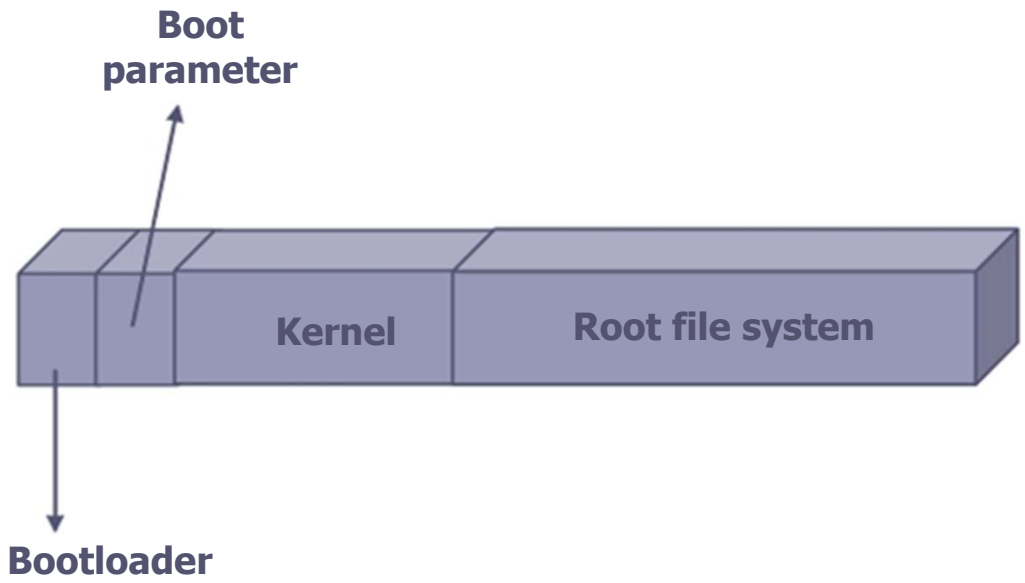
Cài đặt tự động

- **Cài đặt tự động phổ biến khi phát triển xây dựng hệ thông PC mạnh trên cấu trúc của hệ thống nhúng**
  - Có thể sử dụng chuẩn sẵn sàng sử dụng của linux cung cấp cho hệ thống nhúng (off-the-shelf)
  - Một khi đã phát triển xong, hệ thống hoạt động theo trật tự xuống theo sự phân bố và mục đích yêu cầu
- **Đòi hỏi xây dựng một hệ thống gốc filesystems dành cho nó, và được cấu hình khi hệ thống khởi động**

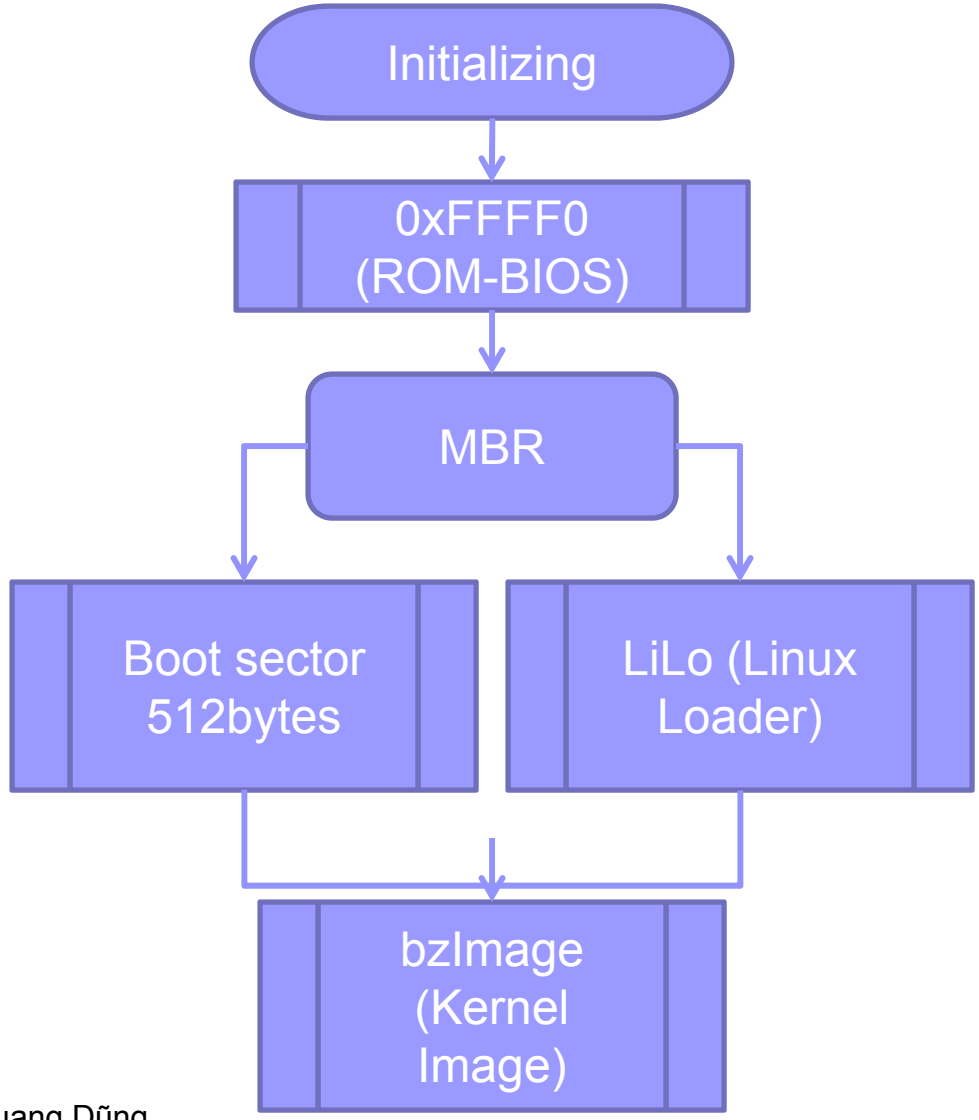
Cài đặt tự động (tiếp)

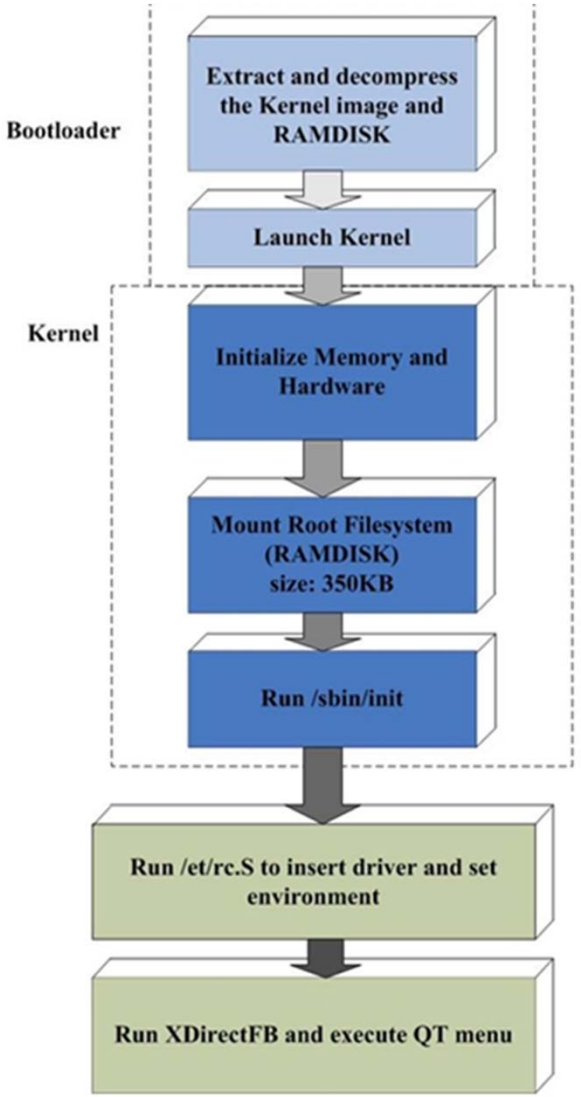
- **Một số đặc điểm riêng biệt từ phần cứng để chạy hệ thống Linux:**
  - **Linux đòi hỏi tối thiểu 32-bit CPU chứa hệ điều khiển bộ nhớ (MMU)**
  - **Đủ dung lượng RAM cần thiết cho hệ thống làm việc**
  - **Năng lực tối thiểu của hệ thống I/O đòi hỏi nếu trong quá trình phát triển cần phải tach riêng đích và nguồn để dễ dàng cho kiểm lỗi**
- **Có 3 sự cài đặt khác nhau được sử dụng để tự khởi động hệ thống linux nhúng:**
  - **Trạng thái lưu trữ vững chắc**
  - **Ổ đĩa**
  - **Mạng**

Trạng thái lưu trữ vững chắc



Ổ đĩa

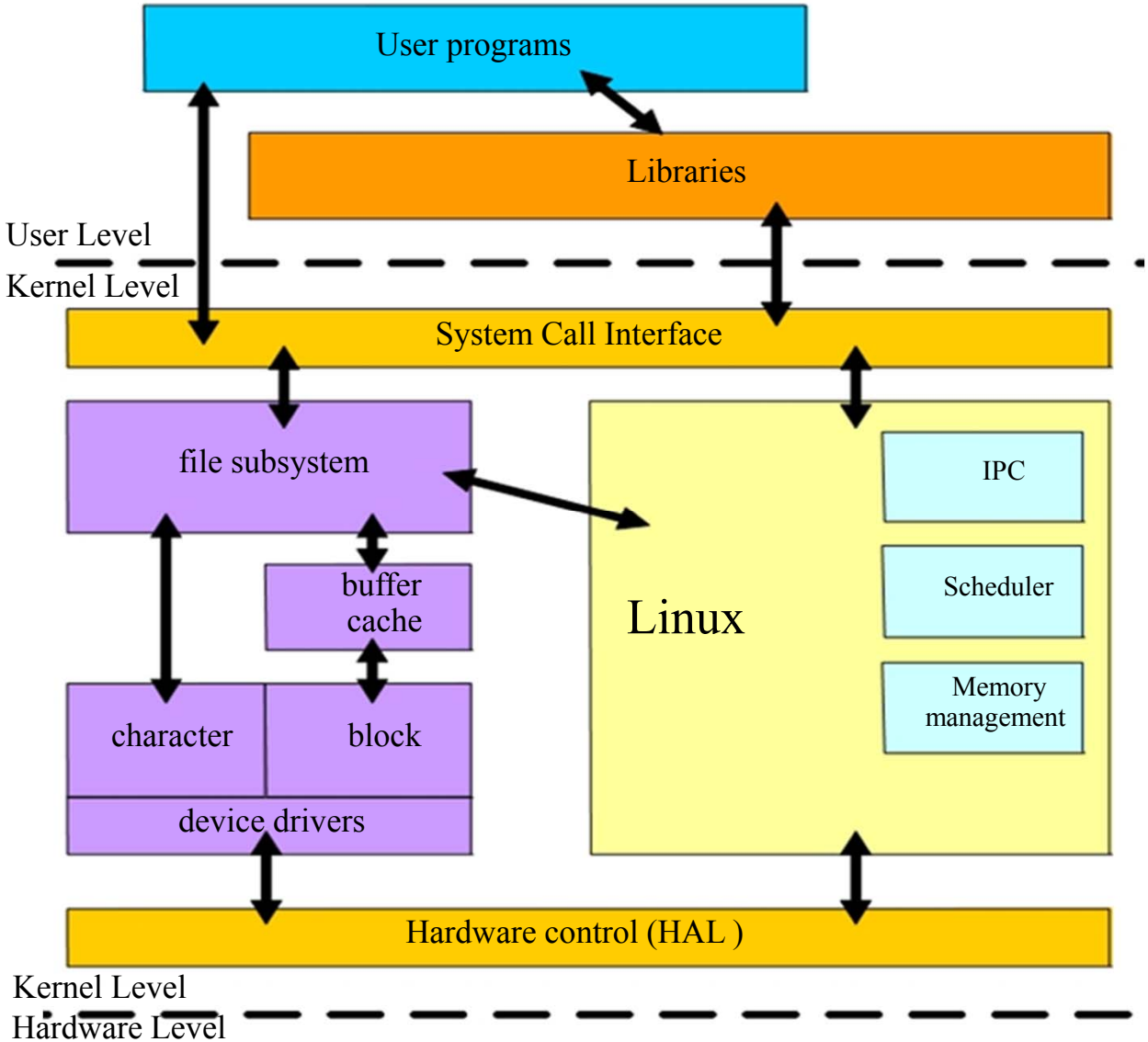

Initializing
0xFFFF0
(ROM-BIOS)
MBR
Boot sector
512bytes
LiLo (Linux
Loader)
bzImage
(Kernel
Image)


Bootloader
Extract and decompress the Kernel image and RAMDISK
Launch Kernel
Kernel
Initialize Memory and Hardware
Mount Root Filesystem (RAMDISK) size: 350KB
Run /sbin/init
Run /et/rc.S to insert driver and set environment
Run XDirectFB and execute QT menu

Mạng

- **Nhân linux tập trung vào trạng thái lưu trữ vững chắc hoặc ổ đĩa, và hệ thống filesystem gốc NFS**
- **Chỉ có bootloader được tập trung vào thiết bị lưu trữ cục bộ. Nhân được cập nhật thông qua TFTP, và hệ thống filesystem gốc NFS**

Nhân linux


User programs
Libraries
User Level
Kernel Level
System Call Interface
file subsystem
buffer cache
character
block
device drivers
Linux
IPC
Scheduler
Memory management
Hardware control (HAL)
Kernel Level
Hardware Level

Điều khiển thiết bị

- **Đặc trưng của thiết bị**
- **Khối thiết bị**
- **Giao diện mạng**

Đặc trưng của thiết bị

- **Đặc trưng cơ bản của thiết bị là có thể truy nhập như là một chuỗi của byte**
- **Sự điều khiển thường để thực thi tối thiểu các chức năng như mở, đóng, đọc, và ghi yêu cầu hệ thống**
- **Giao diện console (/dev/console) và cổng nối tiếp (/dev/ttyS0) là những ví dụ của đặc trưng thiết bị**
- **Đặc trưng của thiết bị là các kênh dữ liệu, mà có thể truy cập liên tiếp**

Khối thiết bị

- **Giống như đặc trưng thiết bị, khối thiết bị là sự truy cập của filesystem vào thư mục /dev.**

Giao diện mạng

- **Bất kỳ sự thực hiện trên mạng được tiến hành thông qua giao diện, thiết bị có thể trao đổi dữ liệu với các host khác**
- **Giao diện mạng có thể được nhìn nhận như hàng đợi (Queue)**
- **Unix cung cấp sự truy cập tới giao diện thông qua tên truy cập riêng danh cho chúng (như eth0).**

Giao diện người sử dụng

- **QT & QT Embedded**
  - **Qt là thư viện lớp C++ dành cho phát triển GUI chạy trên Unix, Windows**
  - **Qt/Embedded là cổng linux nhúng của Qt, là sự hoàn thiện của GUI trên C++ và nền phát triển dành cho Linux nhúng**